AD-A235 464

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

(2)

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		REPRINT VOL 16, NO. 1 March 1990

4. TITLE AND SUBTITLE

AN ADAPTIVE MESH MOVING AND LOCAL REFINEMENT METHOD FOR TIME - DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS

5. FUNDING NUMBERS

AFOSR-85-0156
61102F 2304/A3

6. AUTHOR(S)

DAVID C. ARNEY AND JOSEPH E. FLAHERTY

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180-3590

AFOSR-TR- 91 0478

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFOSR-85-0156

11. SUPPLEMENTARY NOTES

DTIC
ELEC
MAY 07
D
C

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

We discuss mesh-moving, static mesh-regeneration, and local mesh-refinement algorithms that can be used with a finite difference or finite element scheme to solve initial-boundary value problems for vector systems of time-dependent partial differential equations in two space dimensions and time. A coarse based mesh of quadrilateral cells is moved by an algebraic mesh-movement function so as to follow and isolate spatially distinct phenomena. The local mesh-refinement method recursively divides the time step and spatial cells of the moving base mesh in regions where error indicators are high until a prescribed tolerance is satisfied. The static mesh-regeneration procedure is used to create a new base mesh when the existing ones becomes to distorted. The adaptive methods have been combined with a MacCormack finite difference scheme for hyperbolic systems and an error indicator based upon estimates of the local discretization error obtained by Richardson extrapolation. Results are preseentd for several computational exapmles.

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

An Adaptive Mesh-Moving and Local Refinement Method for Time-Dependent Partial Differential Equations

DAVID C. ARNEY
United States Military Academy
and
JOSEPH E. FLAHERTY
Rensselaer Polytechnic Institute

We discuss mesh-moving, static mesh-regeneration, and local mesh-refinement algorithms that can be used with a finite difference or finite element scheme to solve initial-boundary value problems for vector systems of time-dependent partial differential equations in two space dimensions and time. A coarse base mesh of quadrilateral cells is moved by an algebraic mesh-movement function so as to follow and isolate spatially distinct phenomena. The local mesh-refinement method recursively divides the time step and spatial cells of the moving base mesh in regions where error indicators are high until a prescribed tolerance is satisfied. The static mesh-regeneration procedure is used to create a new base mesh when the existing one becomes too distorted. The adaptive methods have been combined with a MacCormack finite difference scheme for hyperbolic systems and an error indicator based upon estimates of the local discretization error obtained by Richardson extrapolation. Results are presented for several computational examples.

Categories and Subject Descriptors: G.1.8 [**Numerical Analysis**]: Partial Differential Equations—*difference methods, hyperbolic equations*; G.4 [**Mathematics of Computing**]: Mathematical Software—*efficiency, reliablity and robustness*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Adaptive methods, local mesh refinement, mesh moving

1. INTRODUCTION

Many initial-boundary value problems for time-dependent partial differential equations involve fine-scale structures that develop, propagate, decay, and/or disappear as the solution evolves. Some examples are shock waves in compressible

This research was partially supported by the U.S. Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant AFOSR 85-0156 and by the SDIO/IST under management of the U.S. Army Research Office under contract DAAL 03-86-K-0112. This research was used to partially fulfill the Ph.D. requirements of D. C. Arney at the Rensselaer Polytechnic Institute.
Authors' addresses: D. C. Arney, Department of Mathematics, United States Military Academy, West Point, NY 10996-1786. J. E. Flaherty, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

flows, boundary and shear layers in viscous flows, and reaction zones in combustion processes. The numerical solution of these problems is usually difficult because the nature, location, and duration of the structures are often not known in advance. Thus, conventional numerical approaches that calculate solutions on a prescribed (typically uniform) mesh often fail to adequately resolve the fine-scale phenomena, have excessive computational costs, or produce incorrect results. Adaptive procedures that evolve with the solution offer a robust, reliable, and efficient alternative. Such techniques have been the subject of a great deal of recent attention (cf. Babuska et al. [8, 9] and Thompson [29]) and are generally capable of introducing finer meshes in regions where greater resolution is needed [1, 2, 3, 6, 14, 15, 22, 26], moving meshes in order to follow isolated dynamic phenomena [1, 2, 5, 20, 22, 26, 28], or changing the order of methods in specific regions of the problem domain [17, 21]. The utility of such adaptive techniques is greatly enhanced when they are capable of providing an estimate of the accuracy of the computed solution. Local error estimates are often used as refinement indicators and to produce solutions that satisfy either local or global accuracy specifications [1, 2, 3, 6, 14, 15, 26]. Successful error estimates have been obtained using h-refinement [6, 14, 15], where the difference between solutions on different meshes is used to estimate the error, and p-refinement [1, 2, 3, 15, 21] where the difference between methods of different orders are used to estimate the error.

We discuss an adaptive procedure that combines mesh movement and local refinement for m-dimensional vector systems of partial differential equations having the form

$$\mathbf{u}_t + \mathbf{f}(x, y, t, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_y) = [\mathbf{D}^1(x, y, t, \mathbf{u})\mathbf{u}_x]_x + [\mathbf{D}^2(x, y, t, \mathbf{u})\mathbf{u}_y]_y, \quad \text{for} \quad t > 0, \quad (x, y) \in \Omega, \tag{1.1a}$$

with initial conditions

$$\mathbf{u}(x, y, 0) = \mathbf{u}^0(x, y), \qquad \text{for} \quad (x, y) \in \Omega \cup \partial\Omega, \tag{1.1b}$$

and appropriate well-posed boundary conditions on the boundary $\partial\Omega$ of a rectangular region Ω.

We suppose that a numerical method is available for calculating approximate solutions and error indicators of (1.1) at each node of a moving mesh of quadrilateral cells. Most numerical methods are applicable, and the error indicator can either be an estimate of the local discretization error or another function (e.g., an estimate of the solution gradient or curvature) that is large where additional resolution is needed and small where less resolution is desired. Our adaptive algorithm consists of three parts: (i) movement of a coarse base mesh, (ii) local refinement of the base mesh in regions where resolution is inadequate, and (iii) creation and regeneration of the base mesh when it becomes overly distorted. Mesh motion can substantially reduce errors of hyperbolic problems for a very modest computational cost. Mesh motion alone, however, cannot produce a solution that will satisfy a prescribed error tolerance in all situations. For this reason, we have combined mesh motion with local mesh refinement and recursively solve local problems in regions where error tolerances are not satisfied. The local solution scheme successively reduces the domain size, and thus further

reduces the cost of the computation. Some problems (e.g., those with severe material deformations) can result in tangling and distortion of the moving base mesh. Therefore, we have created a procedure that automatically generates a new base mesh whenever the old one is unsuitable.

The adaptive procedures described in this paper combine our earlier work on mesh-moving techniques [5] and local refinement procedures [6]. The inclusion of a static mesh-regeneration scheme adds greater reliability and efficiency to these methods. The three components of our adaptive algorithm are described in Section 2; however, frequent references are made to our previous investigations [5, 6]. A computer code based on the adaptive algorithm of Section 2 has been combined with a MacCormack finite difference scheme and an error indicator based on Richardson extrapolation. It has been used to solve a sequence of hyperbolic problems (i.e., problems having the form (1.1) with $\mathbf{D}^1 = \mathbf{D}^2 := 0$) and our findings on three examples, where we have attempted to appraise the relative costs and benefits of the mesh-moving and local refinement portions of our adaptive algorithm, are reported in Section 3. We have also compared solutions obtained by adaptive techniques to those obtained using stationary uniform meshes. Most of the computational time was devoted to calculating the solution and error indicators, and not to the overhead induced by the refinement procedure. Some possible improvements and future considerations are discussed in Section 4.

2. ALGORITHM DESCRIPTION

A top-level description of our adaptive procedure is presented in Figure 1 in a pseudo-Pascal language. This procedure, called *adaptive_PDE_solver*, integrates a system of partial differential equations from time *tinit* to *tfinal* and attempts to keep the local error indicators below a tolerance *tol*. The base-level time step Δt is initially specified, but may be changed as needed during the integration.

The rectangular domain Ω is initially discretized into a coarse moving spatial grid of $M \times N$ quadrilateral cells. An initial base mesh is generated from this mesh by increasing the values of M and N, as necessary, and moving the mesh so that it is concentrated in regions where error indicators are large (cf. Section 2.3). The base mesh is moved for each base time step Δt, as described in Section 2.1, and the partial differential system (1.1) is solved on this mesh for a base time step. This is followed by a recursive local mesh refinement in regions where error indicators are larger than *tol*. The local refinement procedure *local_refine* is described by Arney and Flaherty in [6], and its major features are summarized in Section 2.2. The integration for each base-mesh time step is concluded by the selection of a new value of Δt for the subsequent time step and the generation of new base mesh (cf. Section 2.3), if necessary.

The mesh-moving, local refinement, and mesh-regeneration algorithms are uncoupled from each other as well as from the procedures used to solve the partial differential system and calculate local error indicators. This reduces computational costs and provides a great deal of flexibility.

2.1 Mesh-Moving Algorithm

Mesh-moving strategies should produce a smooth mesh where the sizes of neighboring computational cells vary slowly and cell angles differ only by modest

```
procedure adaptive_PDE_solver(tinit, Δt, tfinal, tol: real; M, N: integer);

  begin
    Generate an initial base mesh;
    t := tinit;

    while t < tfinal do
      begin
        Move the base mesh for the time step t to t + Δt;
        local_refine(0, t, Δt, tol);
        t := t + Δt;
        Select an appropriate Δt;
        if base mesh is too distorted then regenerate a base mesh
      end
  end { adaptive_PDE_solver };
```

Fig. 1. Pseudo-Pascal description of an adaptive procedure to solve the partial differential system (1) from $t = tinit$ to $tfinal$ to within a tolerance of tol.

amounts from right angles. It is, of course, essential for the nodes of the mesh to remain within Ω and for cells not to overlap. Meshes that violate these conditions can produce large discretization errors that overwhelm the positive effects of mesh moving. Our mesh-moving procedure [5] is based on an intuitive approach rather than more analytic error equidistribution [18, 26] and variational approaches [16]. The essential idea is to move the mesh so as to roughly follow isolated nonuniformities, such as wave fronts, shock layers, and reaction zones. This generally reduces dispersive errors and allows the use of larger time steps while maintaining accuracy and stability.

At each base time, we scan the $M \times N$ base mesh of quadrilateral cells and locate "significant error nodes" as those having error indicators greater than twice the mean nodal error indicator and also greater than ten percent of tol. This empirical strategy avoids having the mesh respond to fluctuations when error indicators are too small, but is sensitive enough to avoid missing dynamic phenomena associated with large error indicators. If there are no significant error nodes, computation proceeds on a stationary mesh. The nearest neighbor clustering algorithm of Berger and Oliger [14] is then used to determine the orientation and size of rectangular $w_1 \times w_2$ clusters that contain all of the significant error nodes. One such cluster is shown as the central rectangular region on Figure 2.

Clusters are designed so that nodes within them may be assumed to have related solution characteristics. Thus, we determine mesh movement from the velocity of propagation, the orientation, and the size of each rectangular error cluster. Node movement is determined by the propagation of the center of error of each cluster, which moves according to the differential equation

$$\ddot{\mathbf{r}}_m + \lambda \dot{\mathbf{r}}_m = 0. \tag{2.1a}$$

Here, $\dot{(\ )} := d(\ )/dt$, λ is a scalar parameter, and $\mathbf{r}_m(t) = [x_m(t), y_m(t)]^T$ is the position of the center of error of the cluster, that is,

$$\mathbf{r}_m(t) = \frac{\sum \mathbf{r}_i(t) E_i(t)}{\sum E_i(t)}. \tag{2.1b}$$

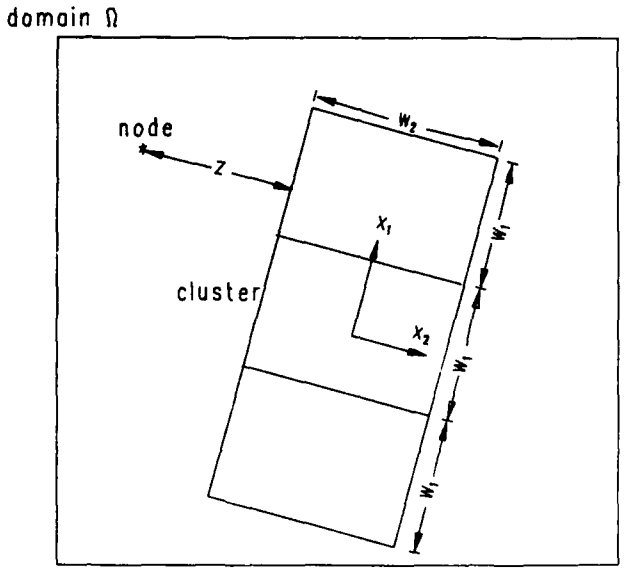

Fig. 2. A rectangular central $w_1 \times w_2$ error cluster. Nodes within the range of the cluster, $3w_1 \times w_2$, are moved a distance $d_{1,inside}$ in the x_1 principal direction according to Eq. (2.2). Nodes outside the range of the cluster are moved a distance $d_{1,outside}$ in the x_1 direction according to Eq. (2.3). The distance z is the shortest distance to the range of the cluster.

The summations range over all nodes in a cluster, $\mathbf{r}_i(t)$ is the position of node i at time t, and $E_i(t)$ is the positive error indicator at node i and time t.

The choice of the parameter λ can be critical in certain situations. If λ is selected too large, the system (2.1) will be stiff and computationally expensive. On the other hand, if λ is too small, the mesh can oscillate from time-step-to-time-step. Coyle et al. [18] and Adjerid and Flaherty [2] suggested some adaptive procedures for choosing λ; however, we found no appreciable differences in results or computation times when λ varied significantly. The examples of Section 3 are calculated with $\lambda = 1$.

We solve (2.1) for each base time step and each cluster using an explicit numerical method. The center of an error cluster is moved a distance $\Delta\mathbf{r}_m = \mathbf{r}_m(t + \Delta t) - \mathbf{r}_m(t)$ at the base time t. Let Δr_{m_1} and Δr_{m_2} denote the projections of $\Delta\mathbf{r}_m$ onto the major and minor axes of the cluster. Mesh motion is performed in two steps, each parallel to a principal axis of an error cluster. To this end, let

$$d_{i,inside} = \begin{cases} \Delta r_{m_i}(\frac{3}{2} + x_i/w_i), & \text{if } -3w_i/2 \le x_i \le -w_i/2 \\ \Delta r_{m_i}, & \text{if } -w_i/2 < x_i < w_i/2 \\ \Delta r_{m_i}(\frac{3}{2} - x_i/w_i), & \text{if } w_i/2 \le x_i \le 3w_i/2 \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, 2, \quad (2.2)$$

and

$$d_{i,outside} = d_{i,inside}[1 - (2z/D)], \qquad i = 1, 2, \tag{2.3}$$

denote one-dimensional piecewise linear functions that move nodes along the two principal axial directions of a $w_1 \times w_2$ cluster and let (x_1, x_2) be local Cartesian coordinates in these principal directions relative to a cluster's center (cf. Figure 2). Consider mesh motion in the x_1 direction. The cluster is enlarged to a region $\{(x, y) \mid 3w_1/2 \le x_1 \le 3w_1/2, -w_2/2 \le x_2 \le w_2/2\}$ called the "range of the cluster" (cf. Figure 2). All nodes in the range of the cluster are moved a distance $d_{1,inside}$ in the x_1 direction. In order to maintain smooth mesh motion throughout the domain, nodes outside the range of a cluster are moved a distance $d_{1,outside}$ in the x_1 direction, where z is the shortest distance to the range of the cluster (cf. Figure 2) and D is the diagonal of Ω.

Motion of all clusters in the directions of their major axes ($i = 1$ in (2.2, 3)) is followed by a similar movement in the directions of their minor axes ($i = 2$ in (2.2, 3)). The distances $d_{i,inside}$ and $d_{i,outside}$ are reduced near $\partial\Omega$ in order to prevent nodes from leaving Ω. In particular, we recalculate $d_{i,j}$ as $d_{i,j}[\min(1, b/c)]$, $i = 1, 2$, $j = inside, outside$, where b is the distance of the node to the boundary and c is twice the length of a cell diagonal on a uniform mesh having the same number of cells as the moving mesh. Nodes on domain boundaries, except corner nodes, which are not moved, are restrained to move along the boundary. Finally, the mesh-moving algorithm is not restricted to the functions given by (2.2) and (2.3), and several other choices are possible.

2.2 Local Refinement Algorithm

As shown in Figure 1, the local refinement procedure is invoked after the base mesh has been moved for a base time step. Our refinement strategy consists of first calculating a preliminary solution on the base mesh for a base time step. An error indicator is used to locate regions where greater resolution is needed. Finer grids are adaptively created in these high-error regions by locally bisecting the time step and the sides of the quadrilateral cells of the base grid, and the solution and error indicators are computed on the finer grids. The refinement scheme is recursive; thus, fine subgrids may be refined by adaptively creating even finer subgrids. This relationship leads naturally to a tree data structure. Information regarding the geometry, solution, and error indicators of the base grid is stored as the root node or level 0 of the tree. Subgrids of the base grid are offspring of the root node and are stored as level 1 of the tree. The structure continues, with a grid at level l having a parent coarser grid at level $l - 1$ and any finer offspring grids at level $l + 1$. Grids at level l of the tree are given an arbitrary ordering, and we denote them as $G_{l,j}$, $j = 1, 2, \ldots, N_l$, where N_l is the number of grids at level l. Our refinement procedures permit grids at the same level of a two-dimensional problem to intersect and overlap; however, offspring grids must be properly nested within the boundaries of their parent grid. A one-dimensional grid with its appropriate tree structure for a base time step is shown in Figure 3.

A top-level pseudo-Pascal description of a recursive local refinement algorithm that solves systems of the form (1.1) on the tree of grids described above is

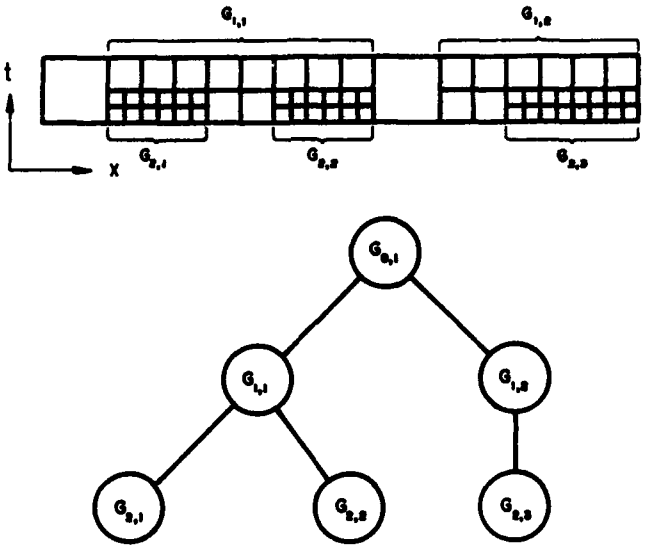

Fig. 3. Coarse and refined grids (top) and their tree representation (bottom) for a one-dimensional example.

presented in Figure 4. The procedure *local_refine* integrates partial differential equations on the grids $G_{l,j}$, $j = 1, 2, \ldots, N_l$, at level l of the tree from time *tinit* to *tinit* + Δt and attempts to satisfy a prescribed local error tolerance *tol*. For each grid at level l, a solution and error indicators are calculated at time *tinit* + Δt. Additional finer grids are introduced in regions where the error indicators exceed the prescribed tolerance *tol*, and the differential system is solved again on the finer grids using two time steps of duration $\Delta t/2$ and a tolerance of *tol*/2. Implicit in *local_refine* are the assumptions that a solution can be computed on any grid and that refinement terminates. If either of these assumptions is violated, the procedure terminates in failure.

Our technique for introducing finer subgrids consists of four steps: (i) an initial scan of each level l grid to locate "untolerable-error" nodes as those where the error indicator exceeds the prescribed tolerance *tol*, (ii) clustering any untolerable nodes into rectangular regions, (iii) buffering the clustered regions in order to reduce problems associated with prescribing initial and boundary conditions at coarse/fine grid interfaces, and (iv) cellularly refining the level l meshes and time step within the buffered clusters. Of course, if there are no untolerable-error nodes, the solution is acceptable and further refinement is unnecessary.

The same clustering algorithm [14] that was used to move the base mesh is also used to group untolerable-error nodes for refinement. Each rectangular error cluster is enlarged by increasing its major and minor axes by twice the size of the average cell edge within the cluster. The region between the enlarged and original error clusters provides a buffer so that artificial internal boundary

```
procedure local_refine( l: integer; tinit, Δt, tol: real );

begin
  for j := 1 to N[l] do
    begin
      Integrate the partial differential system from tinit to tinit + Δt
        on grid G[l,j];
      Calculate error indicators at tinit + Δt at all nodes of grid
        G[l,j];
      if any error indicators > tol then introduce level l + 1 subgrids
        of G[l,j]
    end { for };

  if any error indicators > tol then
    begin
      local_refine(l + 1, tinit, Δt/2, tol/2);
      local_refine(l + 1, tinit + Δt/2, Δt/2, tol/2)
    end
end { local_refine };
```

Fig. 4. Pseudo-Pascal description of a recursive local refinement procedure to find a solution to the partial differential system (1.1) on all grids at level l of the tree.

conditions (discussed below) will be prescribed at low-error nodes as far as possible and fine-grid errors will not propagate through the buffer in a time step.

Refined subgrids are created by bisecting the time step and edges of each cell of the parent mesh that intersects the buffered rectangular error clusters. Coarse mesh motion is maintained on the refined grids so that, after two time steps of size $\Delta t/2$, cells of the refined grids will be properly nested within those of their parent grid. Additional details of the refinement algorithm and data structures are presented by Arney and Flaherty in [6].

Artificial initial and boundary data must be determined from solutions on other grids in order to calculate the solution and error indicators on refined subgrids. Initial data for a subgrid are calculated directly from the initial function $\mathbf{u}^0(x, y)$ at $t = 0$. For $t > 0$, initial data is obtained by interpolation using the solution at the same time on the finest available mesh. In order to provide data for this interpolation, we save all solution values on previous subgrids until they are no longer needed due to advancement in time of an acceptable solution. Bilinear functions using the solution values at the four vertices of the finest existing cell are used to obtain the solution at the nodes of cells of the refined mesh. Further analysis is needed regarding the effects on accuracy and stability and the proper order of this interpolation. Bieterman, Flaherty, and Moore [15] give an example where the fine-scale structure of a solution was lost by interpolation from too coarse a mesh.

In a similar manner, boundary data for refined meshes are calculated directly from the prescribed boundary conditions on portions of subgrids that intersect $\partial\Omega$. Dirichlet boundary data are prescribed on the edges of subgrids that are in the interior of Ω by interpolating the solution from coarser meshes. Bilinear functions using the solution values at the four vertices of the adjacent face of the

finest existing space-time cell are used to obtain solution values for the nodes of refined cells.

Acceptable fine-mesh solutions are used to replace solutions at the nodes of coarser grids that lie within the untolerable-error portions of clusters. Solutions at low-error nodes in the buffer zones of clusters are not replaced in order to avoid possible contamination of accurate solutions. When fine grids overlap each other in an untolerable-error region, the average value of the solutions at common fine-grid nodes is used to replace the appropriate coarse grid solution. Boundary effects do not propagate through a sufficiently large buffer, and thus have no effect on the solution within the untolerable-error region of a cluster when an explicit numerical scheme is used for the integration. Greater care is needed when implicit integration methods are used, since artificial boundary conditions can affect the accuracy, convergence, and stability of the solution at all nodes in the cluster regardless of the size of the buffer.

Stability and conservation of, for example, fluxes at interfaces between coarse and fine meshes must be investigated further, particularly in two dimensions. For one-dimensional problems, Berger and Oliger [14] showed that linear interpolation of solutions from a coarse to a fine mesh produced no instabilities in the Lax–Wendroff scheme. Berger [12] also discussed conservation at mesh interfaces and proposed explicit enforcement of conserved quantities at coarse/fine mesh boundaries. Rai [27] presented some finite difference schemes that maintained conservation at grid interfaces for two-dimensional compressible flow problems.

2.3 Initial Mesh Construction and Regeneration

The efficiency of our adaptive mesh-moving and refinement strategies are dependent on our ability to generate a suitable initial mesh and to regenerate a new base mesh, should it become severely distorted at later times. The proper base mesh can reduce the need for refinement, and thus increase efficiency.

The two essential elements of a mesh-generation or regeneration procedure are the determination of the number of nodes and their optimal location. A base mesh having too few nodes will result in excessive refinement, while one having too many nodes will reduce efficiency. Our approach to mesh generation is to use the error indicators computed by a trial solution to determine an initial mesh that approximately equidistributes the error indicators.

To begin, we create a uniform $M \times N$ rectangular mesh using prescribed values of M and N that reflect the coarsest mesh that should be used to calculate a solution. We solve the system (1.1) for a base time step Δt on the uniform stationary base mesh and compute the solution and error indicators. Local mesh refinement is performed as described in Section 2.2 until the prescribed tolerance is attained. We use this solution to determine the number of nodes K in a new base mesh as

$$K = MN + \sum_{l=1}^{n} \left(\tfrac{3}{4}\right)^l K_l, \tag{2.4a}$$

where K_l is the number of nodes introduced at level l and n is the total number of levels in the tree. Having computed K, we calculate the dimensions

of a new $\bar{M} \times \bar{N}$ mesh as

$$\bar{M} = \sqrt{KM/N}, \qquad \bar{N} = \sqrt{KN/M}. \tag{2.4b}$$

The assumption expressed by (2.4a) is that three-fourths of the nodes introduced by refinement at level l are actually required to achieve the prescribed accuracy. The remaining nodes are associated with the buffers and those sharing common locations with level $l-1$ meshes. The assumption expressed by (2.4b) is that the new base mesh should have K nodes distributed in proportion to the original base mesh. (The bars have been omitted on M and N in the algorithms displayed in Figures 1 and 4 and in all further discussions.)

Node placement for the new base mesh is accomplished by locating all nodes of the original base mesh having error indicators that are greater than twice the mean error indicator. These nodes are then grouped into rectangular clusters and are moved toward the center of the nearest error cluster by a procedure similar to the one described in Section 2.1. Additional details are presented by Arney and Flaherty in [7].

A new base mesh can be generated whenever the existing one becomes severely distorted. Since this new mesh is created at a specific time, rather than by mesh motion, we refer to this process as *static mesh regeneration*. Our static mesh-regeneration procedure consists of three steps: (i) determining that there is a need for a new base mesh, (ii) creating the new base mesh, and (iii) interpolating the solution from the old to the new base mesh.

A mesh is regenerated when any interior angle of a cell is less than 50 or greater than 130 degrees, the aspect ratio of any cell is greater than 15, or the mesh ratio of adjacent cells exceeds 5 or is less than $\frac{1}{5}$. In the present context, the aspect ratio is defined as the average length divided by the average width of a cell, and the mesh ratios are defined as the ratio of the lengths and widths of adjacent cell sides.

A new base mesh, having the same number of nodes as the old one, is generated using the procedure described above for creating an initial base mesh. The error clusters for the existing mesh are used to generate the new base mesh, so that new clusters do not have to be computed. This process appears to reduce angle deviations from ninety degrees, control aspect ratios, and mollify adjacent mesh ratios.

Once a new base mesh has been constructed, the solution on the old one is interpolated to the new one by using bilinear interpolation with respect to the cells of the old base mesh. This procedure is inadequate and, at the very least, should use fine as well as coarse grid data for the interpolation. The order and nature of the interpolation also needs further investigation, and we are studying methods that, for example, conserve fluxes (cf. Berger [12] or Rai [27]).

3. COMPUTATIONAL EXAMPLES

In order to demonstrate the capabilities of the adaptive procedure described in Section 2, we applied it to three hyperbolic systems. We used a two-step MacCormack finite difference method (cf. Arney and Flaherty [5], Hindman [23], or MacCormack [24]) to integrate the partial differential equations and Richardson's extrapolation (cf. Arney et al. [4]) to indicate local errors. This

error indication procedure differs from Berger and Oliger's [14] in that (i) separate estimates of the temporal and spatial errors are produced in order to reduce computation and provide the possibility of using differential refinement in different regions of the problem domain, and (ii) an error estimate is obtained on the finest grid. Base mesh geometry was prescribed as indicated in each example. If the base mesh time step failed to satisfy the Courant, Friedrichs, Lewy criterion, it was automatically reduced to the maximum allowed by the Courant condition (cf. Arney et al. [4, 6]). This procedure should also satisfy the Courant condition on all subgrids when the characteristic speeds vary slowly.

Numerical results obtained on uniform stationary grids are compared with those obtained by adaptive strategies that use (i) mesh moving only, (ii) local refinement only, and (iii) the combination of mesh moving and refinement discussed in Section 2. The examples are designed to determine the relative cost, accuracy, and efficiency of our adaptive algorithm and each of its components. Accuracy is appraised by computing the difference $\mathbf{e}$ between the exact and numerical solutions of a problem in either the maximum or L_1 norms, that is, by computing either

$$\| \mathbf{e}(\cdot, \cdot, t) \|_\infty := \max_{1 \le i \le K} \max_{1 \le j \le m} |e_j(x_i, y_i, t)|, \tag{3.1a}$$

or

$$\| \mathbf{e}(\cdot, \cdot, t) \|_1 = \int\int_\Omega P \sum_{j=1}^{m} |e_j| \, dx \, dy, \tag{3.1b}$$

respectively. Here, K is the number of nodes in the mesh at time t, and P is a piecewise constant interpolation operator with respect to the cells of the base mesh that, on each cell, has the average value of the errors at the vertices of the cell. We use either the total CPU time or the maximum number of nodes used in a base time step as measures of the computational complexity of a procedure. The CPU times of calculations performed on uniform stationary meshes also include error estimation times, since error estimation is an integral part of the solution algorithm and adds reliability to the results. All calculations were performed in double-precision arithmetic on an IBM 3081/D computer at the Rensselaer Polytechnic Institute.

Solutions are displayed by drawing either level lines or wire-frame perspective renditions. Meshes are displayed by showing the complete two-dimensional spatial discretization at specified times with finer subgrids overlaying coarser ones. The broken-line rectangles in Figures 5, 6, 9, and 11 indicate the cluster(s) that are used to move the base mesh.

Example 1. Consider the linear initial-boundary value problem introduced by McRae et al. [25] and frequently used as an example [14, 22]:

$$u_t - yu_x + xu_y = 0, \qquad t > 0, \quad (x, y) \in \Omega, \tag{3.2a}$$

$$u(x, y, 0) = \begin{cases} 0, & \text{if } (x - \frac{1}{2})^2 + 1.5y^2 \ge \frac{1}{16} \\ 1 - 16((x - \frac{1}{2})^2 + 1.5y^2), & \text{otherwise,} \end{cases}$$

$$(x, y) \in \Omega \cup \partial\Omega, \tag{3.2b}$$

Table I. Errors at $t = 3.2$ and Computational Costs for Five Solutions of Example 1

Reference No.	Strategy	Base mesh	Base time step	$\|e\|_1$	$\|e\|_\infty$	CPU time (sec)
1	Stationary uniform mesh	14×14	0.056	0.2560	0.78	46
2	Moving mesh	32×32	0.026	0.0301	0.20	458
3	Stationary mesh with refinement	14×14	0.056	0.0832	0.48	852
4	Moving mesh with refinement	14×14	0.056	0.0249	0.18	904
5	Stationary uniform mesh	56×56	0.014	0.0759	0.48	2647
6	Stationary uniform mesh	84×84	0.0093	0.0257	0.29	8670

and

$$u(x, y, t) = 0, \qquad t > 0, \quad (x, y) \in \partial\Omega, \tag{3.2c}$$

where $\Omega := \{(x, y) \mid -1.2 < x, y < 1.2\}$.

The exact solution of (3.2) is an elliptical cone that rotates about the origin in the counterclockwise direction with period 2π. In can be written in the form

$$u(x, y, t) = \begin{cases} 0, & \text{if } C < 0 \\ C, & \text{if } C \geq 0, \end{cases} \tag{3.3a}$$

where

$$C = 1 - 16[(x\cos t + y\sin t - \tfrac{1}{2})^2 + 1.5(y\cos t - x\sin t)^2]. \tag{3.3b}$$

Six adaptive and uniform mesh solutions of (3.2) were calculated for $0 < t \leq 3.2$; our findings are summarized in Table I. Solutions 3 and 4, with refinement, were calculated using an error tolerance of 2×10^{-4} and a maximum of two levels of refinement. The tolerance and maximum level of refinement were selected so that the high-error region under the cone would maintain approximately the same mesh spacing as the uniform mesh used to obtain Solution 5. The grids used to obtain Solution 4 are shown in Figure 5 at $t = 0.56$, 1.68, 2.24, and 3.2. A new base mesh was introduced at $t = 2.82$. The meshes used to obtain Solutions 2, 3, and 4 at $t = 3.2$ are shown in Figure 6. Finally, surface and contour plots of Solutions 1, 2, and 3 and of Solutions 4 and 5 at $t = 3.2$ are shown in Figures 7 and 8, respectively.

Solution 1 bears no resemblance to the exact solution and demonstrates the devastating effects of large dissipative and dispersive errors. Solution 2, with mesh moving only, provides a dramatic improvement in the results for approximately one-half the cost of using both mesh motion and refinement. The subgrids for the refined Solutions 3 and 4 are concentrated in the region of the cone and are aligned with its principal axes as it rotates. Dissipative and dispersive errors cause a "wake" of spurious oscillatory information to follow the moving cone (cf. Figures 7 and 8). Some mesh refinement is performed in the wake region, and this greatly reduces the magnitude of the oscillations.

Solutions 4 and 6 have approximately the same errors in L_1. The adaptive procedure (Solution 4) took less than 11 percent of the computational time of the fixed mesh calculation (Solution 6). Solutions 3 and 5 also have approximately the same errors in both the L_1 and L_∞ norms. This was by design, since both

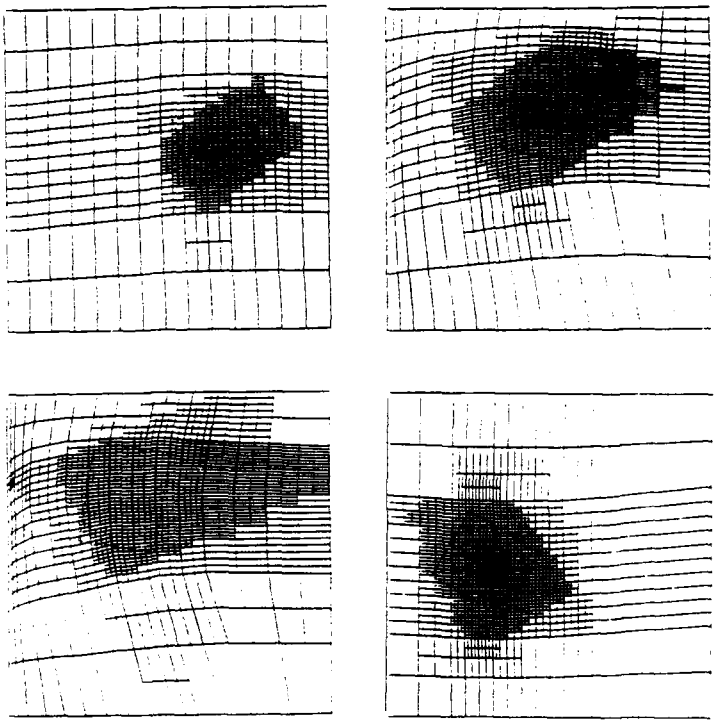

Fig. 5. Grids created for Solution 4 of Example 1 at $t = 0.056$ (upper left), 1.68 (upper right), 2.24 (lower left), and 3.2 (lower right).

solutions have the same mesh beneath the rotating cone. The local refinement method (Solution 3) took approximately 33 percent of the time of the fixed mesh calculation (Solution 5). Berger and Oliger [14] compared solutions of this problem on a fixed uniform mesh and one using one level of local refinement with a four-to-one ratio between the coarse and fine grids. They found that the adaptive local refinement procedure took 16 percent of the time of the fixed mesh computation having the same accuracy. This is slightly less efficient than our combined moving and refinement method (Solution 4), but more efficient than our local refinement procedure (Solution 3). The latter differences are due (i) to our two levels of binary refinement compared to their one level of fourfold refinement and (ii), more significantly, to our use of the error estimate on the finest rather than the coarsest mesh. Our error estimation policy is clearly more expensive, but adds greater reliability to the computed solution.

Example 2. Consider the uncoupled linear initial-boundary value problem:

$$u_{1_t} + u_{1_x} = 0, \qquad u_{2_t} - u_{2_x} = 0, \qquad t > 0, \quad (x, y) \in \Omega, \tag{3.4a}$$

$$u_1(x, y, 0) = \begin{cases} 1 - 16((x - \frac{1}{2})^2 + 1.5y^2), & \text{if } (x - \frac{1}{2})^2 + 1.5y^2 \le \frac{1}{16} \\ 0, & \text{otherwise,} \end{cases}$$

$$(x, y) \in \Omega \cup \partial\Omega, \tag{3.4b}$$

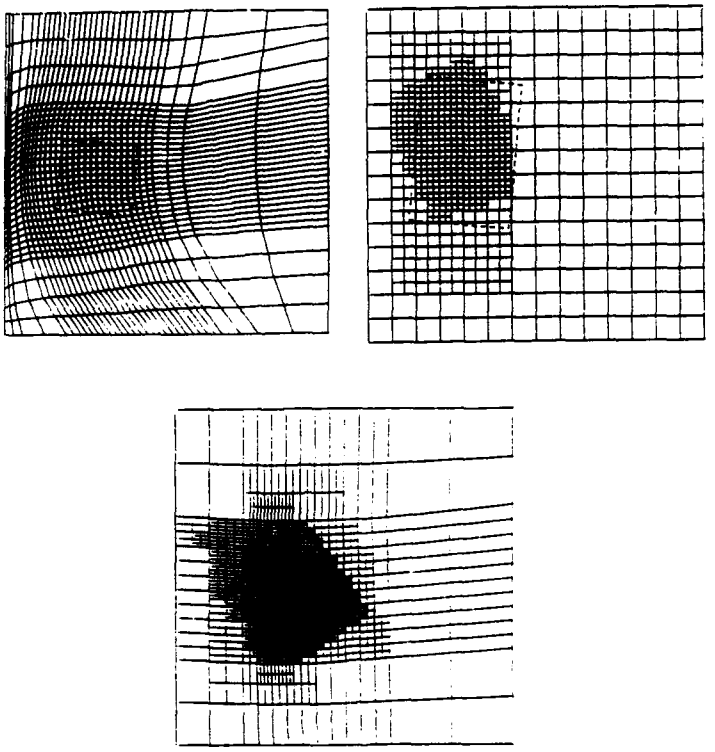

Fig. 6. Grids created for Solutions 2, 3, and 4 of Example 1 at $t = 3.2$.

$$u_2(x, y, 0) = \begin{cases} 1 - 16((x + \frac{1}{2})^2 + 1.5y^2), & \text{if } (x + \frac{1}{2})^2 + 1.5y^2 \le \frac{1}{16} \\ 0, & \text{otherwise,} \end{cases}$$

$$(x, y) \in \Omega \cup \partial\Omega, \quad (3.4c)$$

$$u_1(x, y, t) = u_2(x, y, t) = 0, \qquad t > 0, \quad (x, y) \in \partial\Omega, \qquad (3.4d)$$

and $\Omega := \{(x, y) \mid -1 \le x \le 1, -0.6 \le y \le 0.6\}$.

The solution of this problem consists of two moving cones that collide and pass through each other. We selected it in order to determine how the various adaptive strategies could cope with interacting phenomena.

One uniform mesh and three adaptive solutions of (3.4) were calculated for $0 < t \le 1.2$; our findings are summarized in Table II. The solutions involving refinement were computed with a tolerance of 0.0038. All solutions were designed to have approximately the same accuracy. The grids that were used to obtain Solution 4 are shown in Figure 9 at $t = 0$, 0.23, 0.46, 0.92, and 1.2.

The results of Table II demonstrate the efficiency of mesh moving. We suspect that the accuracy achieved by mesh moving on this example is due to the reduction

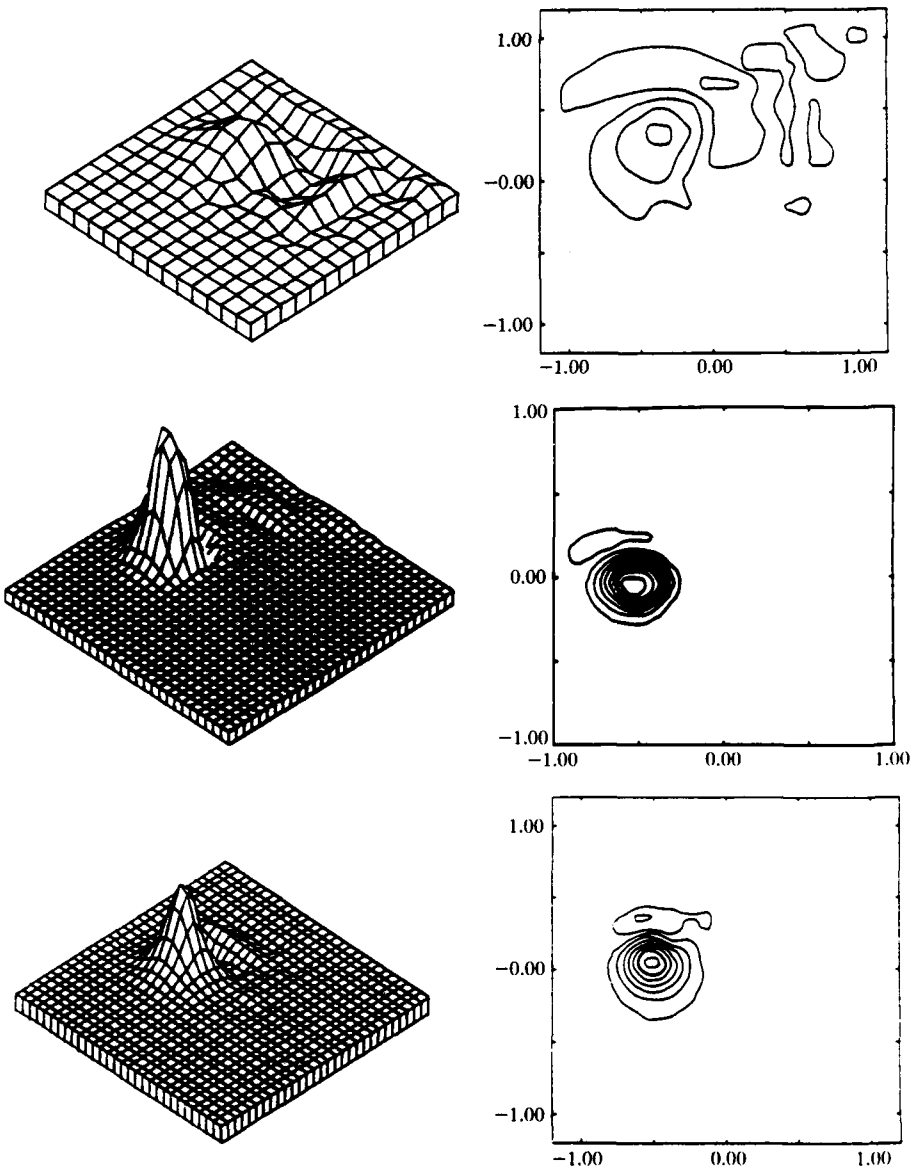

Fig. 7. Surface and contour plots for Solutions 1, 2, and 3 (top to bottom) at $t = 3.2$ of Example 1.

in dispersive errors that occur during the time steps when the mesh follows the cones with approximately the correct velocity. Solution 3 with refinement on a stationary mesh shows only a modest improvement over Solution 5; however, the combination of mesh moving and refinement computed in Solution 4 again shows

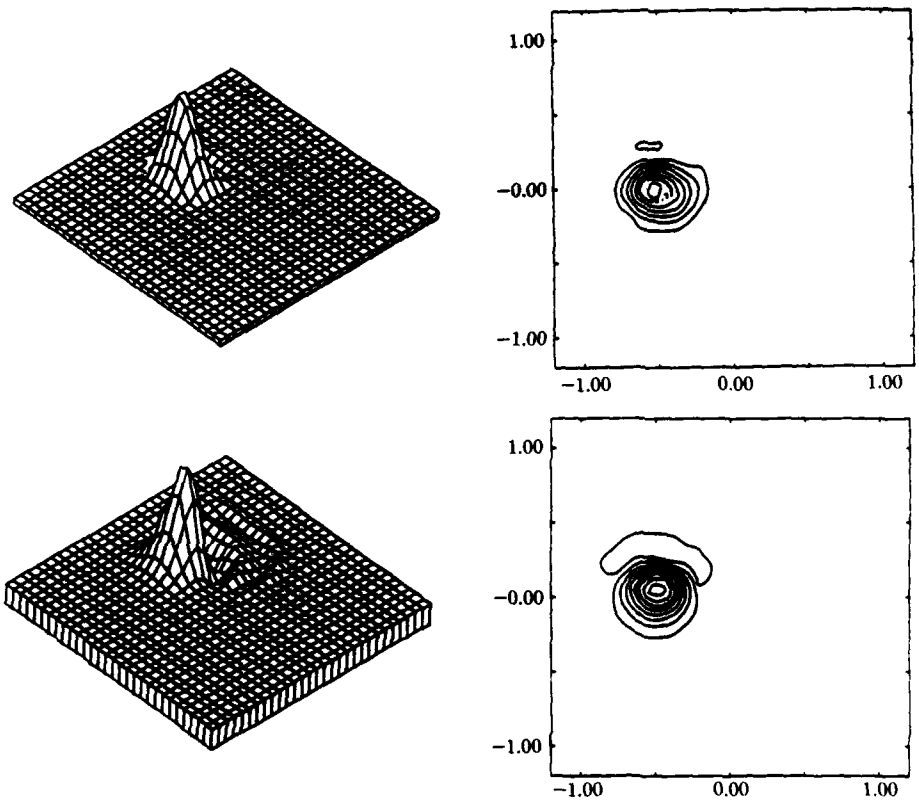

Fig. 8. Surface and contour plots for Solutions 4 (top) and 5 (bottom) at $t = 3.2$ of Example 1.

Table II. Errors at $t = 1.2$ and Computational Costs for Four Solutions of Example 2

Reference No.	Strategy	Base mesh	$\|e\|_1$	$\|e\|_\infty$	CPU time (sec)
1	Stationary uniform mesh	44 × 20	0.099	0.31	269
2	Moving mesh	44 × 20	0.056	0.18	340
3	Stationary mesh with refinement	44 × 20	0.055	0.23	719
4	Moving mesh with refinement	44 × 20	0.039	0.16	609
5	Stationary uniform mesh	64 × 34	0.066	0.26	710
6	Stationary uniform mesh	88 × 40	0.040	0.15	2165

a significant gain in accuracy. Solutions 4 and 6 have similar accuracy in L_1; however, the adaptive procedure (Solution 4) costs 28 percent of the time of the fixed mesh calculation (Solution 6). This ratio is higher than was observed for Example 1. This is most likely due to lack of mesh movement while the solution structures interact and the two error clusters unite to form a single stationary cluster. A similar loss of efficiency was reported by Gropp [22].

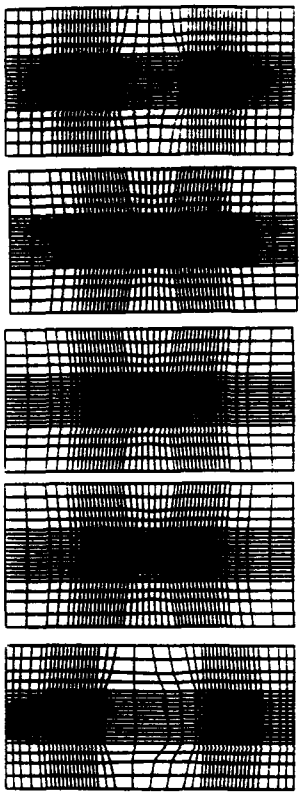

Fig. 9. Grids created for Solution 4 of Example 2 at $t = 0$, 0.23, 0.46, 0.92, and 1.2 (top to bottom).

Example 3. Consider the Euler equations for a perfect inviscid compressible fluid:

$$\mathbf{u}_t + \mathbf{f}_x(\mathbf{u}) + \mathbf{g}_y(\mathbf{u}) = 0, \tag{3.5a}$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{bmatrix}, \quad \mathbf{g}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{bmatrix}. \tag{3.5b, c, d}$$

Here, u and v are the velocity components of the fluid in the x and y directions, ρ is the fluid density, e is the total energy of the fluid per unit volume, and p is the fluid pressure. For an ideal gas:

$$p = (\gamma - 1)[e - \rho(u^2 + v^2)/2], \tag{3.5e}$$

where γ is the ratio of the specific heat at constant pressure to that at constant volume.

We solve a problem where a Mach 10 shock in air ($\gamma = 1.4$) moves down a channel containing a wedge with a half-angle of thirty degrees. This problem was used by Woodward and Collela [30] to compare several finite difference schemes on uniform grids. Like them, we orient a rectangular computational domain, $-0.3 \le x \le 3.4$, $0 \le y \le 1$, so that the top edge of the wedge is on the bottom of the domain in the interval $y = 0$, $\frac{1}{6} \le x \le 3.4$. Thus, in the computational domain, it appears as though a Mach 10 shock is impinging on a flat plate at an angle of sixty degrees. The initial conditions that are appropriate for this situation are the following:

$$\rho = 8.0, \quad p = 116.5, \quad e = 563.5, \quad u = 4.125\sqrt{3}, \quad v = -4.125, \quad \text{if } y < \sqrt{3}(x - \tfrac{1}{6}), \tag{3.6a}$$

and

$$\rho = 1.4, \quad p = 1.0, \quad e = 2.5, \quad u = 0, \quad v = 0, \quad \text{if } y \ge \sqrt{3}(x - \tfrac{1}{6}). \tag{3.6b}$$

Along the left boundary ($x = -0.3$) and the bottom boundary to the left of the wedge ($y = 0$, $-0.3 \le x \le \frac{1}{6}$), we prescribe Dirichlet boundary conditions according to (3.6); along the top boundary ($y = 1$), values are prescribed that describe the exact motion of an undisturbed Mach 10 shock; along the right boundary ($x = 3.4$), all normal derivatives are set to zero; and along the wedge ($y = 0$, $\frac{1}{6} \le x \le 3.4$) reflecting boundary conditions are used.

The solution of this problem is a complete self-similar structure called a double-Mach reflection that was described in Ben-Dor and Glass [10, 11]. Two reflected Mach shocks form with their associated Mach stems and contact discontinuities. The geometry of these structures is very fine, and primarily confined to a small region that moves along the wedge with the incident shock. One of the two contact discontinuities is so weak that it is usually not noticed in computations.

The MacCormack finite difference scheme needs artificial viscosity to "capture" shocks without excessive oscillations. We used a model developed by Davis [19], which is total variation, diminishing in one space dimension. This computational strategy is not competitive with the higher-order methods considered by Woodward and Collela [30] and Berger and Collela [13]. The latter approach used a local refinement procedure, which additionally restricted the problem domain to regions of interest during higher levels of refinement. They thereby obtained results that advance the state of the art for interacting shock phenomena. Our goal was to make some preliminary judgements about the suitability of the various adaptive strategies without tailoring methods for specific applications.

Five solutions of this problem were calculated for $0 < t \le 1.9$ as indicated in Table III. Refinement was restricted to a maximum of two levels, and a tolerance of 0.6 in the maximum norm was prescribed. A pointwise error indicator based on the assumption of smooth solutions, like the present one, is not appropriate for problems having discontinuities. Without restricting the maximum level of refinement, we could refine indefinitely in the vicinity of a discontinuity.

Table III. Maximum Number of Nodes in Any Base Time Step and Computational Costs for Five Solutions of Example 3

Reference No.	Strategy	Base mesh	Maximum no. nodes	CPU time (sec)
1	Stationary uniform mesh	63 × 29	1827	2130
2	Moving mesh	63 × 29	1827	2220
3	Stationary mesh with refinement	29 × 11	2782	3254
4	Moving mesh with refinement	29 × 11	3540	3725
5	Stationary uniform mesh	120 × 40	4800	6861

Solutions 2 through 5 were intended to be of comparable accuracy, and we shall attempt to appraise the computational cost of each adaptive strategy. Contours of the density at $t = 0.19$ are shown for all five solutions in Figure 10, and the grids that were generated for Solution 4 at $t = 0.038$, 0.076, 0.114, 0.152, and 0.19 are shown in Figure 11.

As in the previous two examples, the mesh-moving strategy of Solution 2 does a great deal to improve the results of the static Solution 1. Comparing the top two contours of Figure 10, we see that the resolution of the incident and reflected shocks is much finer with Solution 2 than with Solution 1. Additional details of the structures in the Mach stem region and of the contact discontinuities are present in Solution 2, but not in the nonadaptive Solution 1. Finally, Solutions 1 and 5 display more oscillatory behavior behind the incident shock near the upper boundary. This is undoubtedly due to our maintaining a discontinuity where the shock intersects the upper boundary.

The use of refinement on a stationary mesh again does not give the dramatic improvement obtained by mesh moving (cf. the second and third contours of Figure 10). Initially, the fine meshes followed the incident and reflected shock structures, and better results were obtained; however, by $t = 0.19$, refinement is being performed over much of the domain, and two levels of refinement are not sufficient for adequate resolution (cf. Arney and Flaherty [6]). The combination of mesh motion and refinement depicted by Solution 4 in Figure 10 provides a marked improvement in resolution. The sequence of meshes shown in Figure 11 shows that the coarse mesh is able to follow the differing dynamic structures and that refinement is only performed in the vicinity of discontinuities. Initially, only one rectangular cluster was needed to follow the incident shock (cf. Arney and Flaherty [5]). As time progresses, two clusters are created in order to follow the incident and reflected shocks (cf. the upper three meshes of Figure 11). A third cluster is created as time increases further, in order to follow the evolving activity in the region of the Mach stem (cf. the lower two meshes of Figure 11).

Severe distortion of the mesh in the reflected shock region caused a static mesh regeneration to occur for Solution 4 at $t = 0.162$. The base meshes before and after the static regeneration are shown in Figure 12. Thus, Solution 4 demonstrates all of the capabilities of our adaptive procedure. Solution 4 also shows many of the same characteristics as the solution computed by Woodward and Collela [30] using MacCormack's method on a 240 × 120 uniform grid. We were unable to compute a solution on such a fine mesh due to virtual memory

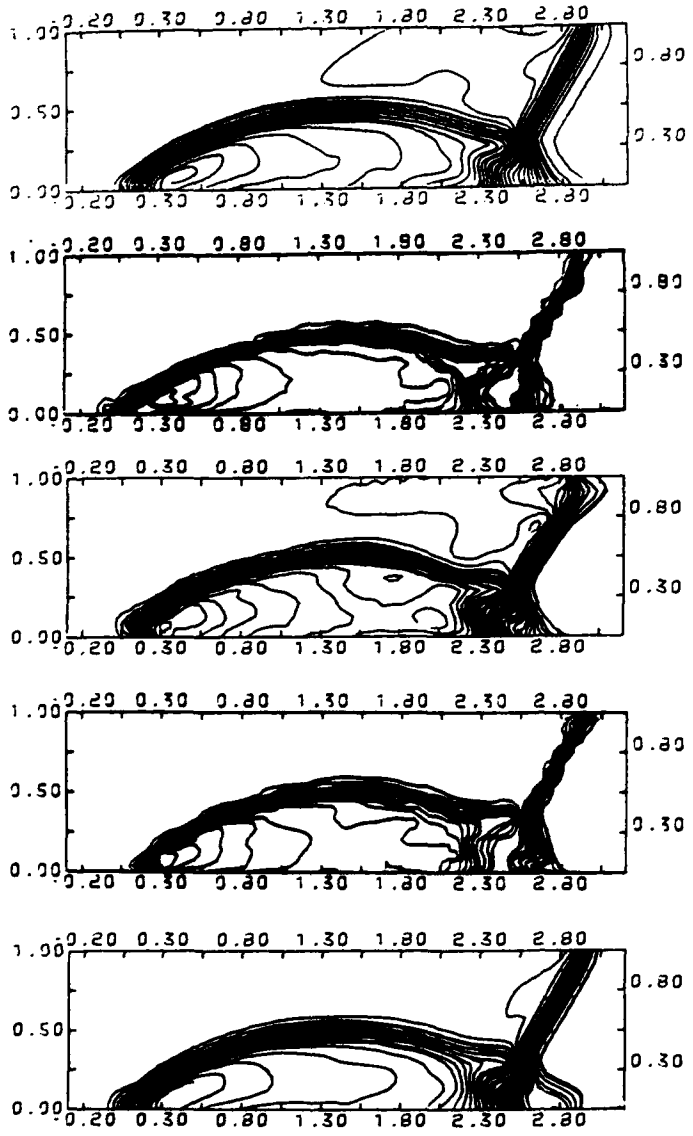

Fig. 10. Contours of the density at $t = 0.19$ for Solutions 1 to 5 (top to bottom) of Example 3.

limitations on our computer; however, we estimate that it would have used 14,400 nodes and 40,000 CPU seconds.

The results presented for this problem demonstrate the power and efficiency of our adaptive techniques; however, we would have preferred to allow more than

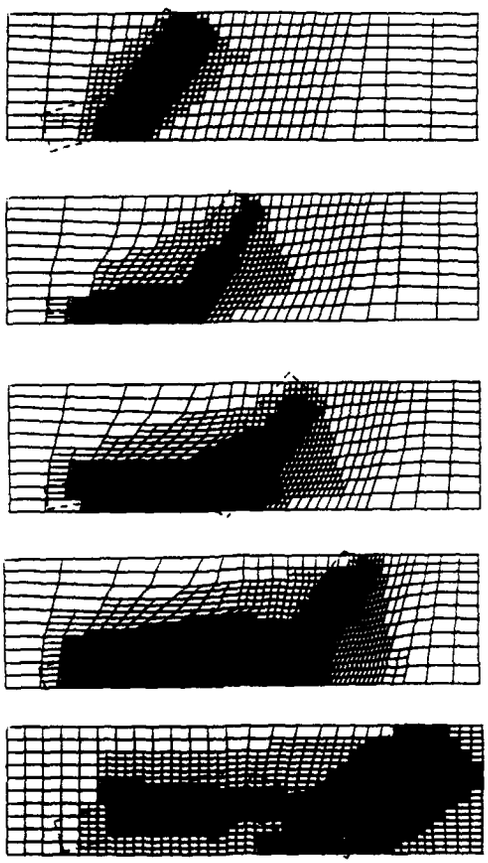

Fig. 11. Grids created for Solution 4 of Example 3 at $t = 0.38$, 0.076, 0.114, 0.152, and 0.19 (top to bottom).

two levels of refinement and a finer base mesh. These calculations would have produced better resolution of the discontinuities and other fine-scale structures that further demonstrate the computational advantages of adaptive methods relative to uniform mesh techniques. As noted, restrictions of our computing environment prevented us from doing this in a reasonable manner. We hope to perform these calculations in the future using a larger computing system.

4. DISCUSSION OF RESULTS AND CONCLUSIONS

We have described an adaptive procedure for solving systems of time-dependent partial differential equations in two-space dimensions that combines existing mesh-moving [5] and local refinement [6] techniques. The algorithm also contains procedures for initial mesh generation and static mesh regeneration. It can be used with a wide variety of finite difference or finite element schemes and error indicators.

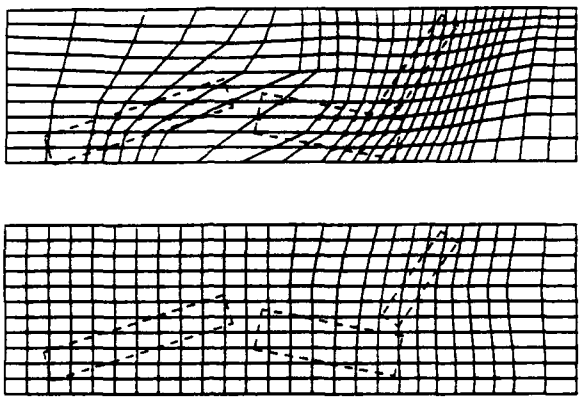

Fig. 12. Base grids before (top) and after (bottom) the static mesh regeneration that was performed for Solution 4 of Example 3 at $t = 0.162$.

We obtained computational results for hyperbolic systems of conservation laws by using our adaptive methods with a MacCormack finite difference scheme and by using Richardson's extrapolation to furnish local error indicators. Our computational results on three examples indicate that mesh moving can significantly reduce errors. The use of local refinement without mesh moving provided increased efficiency relative to uniform-mesh calculations, although not as dramatic as that found using mesh moving.

The results of Section 3 and Arney and Flaherty [5, 6] indicate that our mesh-moving procedures perform better alone than with refinement. This is because the projection of fine-mesh solutions onto coarser meshes reduces the errors at base mesh nodes and mesh motion based on controlling small or zero local discretization errors either fails or results in no movement. Erratic mesh motion can also occur with some techniques when movement indicators are small. This topic is discussed in Coyle et al. [18], and a possible remedy for one-dimensional problems is suggested by Adjerid and Flaherty [2]. Further experimentation and analysis are being performed in order to determine the best way to combine mesh moving and refinement.

There are several other ways to improve the efficiency, reliability, and robustness of our adaptive methods. MacCormack's finite difference procedure could be replaced with other more appropriate methods for shock computation [13, 17, 30]. The present Richardson's extrapolation-based error indicator is expensive, and we are seeking ways of replacing it by techniques using p-refinement. Such methods have been shown [1, 2, 3, 15, 21] to have an excellent cost performance ratio when used in conjunction with finite element methods. An appropriate error indicator or estimator can be used to control a differential refinement algorithm, where different refinement factors (i.e., other than binary) are used in different high-error clusters. If the error indicator is capable of providing separate estimates of the spatial and temporal errors, as the present one does, then different

refinement factors can also be used in space and time. We also hope to demonstrate the flexibility of our refinement procedure by using it with a finite difference or finite element scheme for parabolic problems.

The greater reliability and efficiency of adaptive techniques will be most beneficial in three dimensions. These techniques must be able to take advantage of the latest advances in vector and parallel computing hardware. The tree is a highly parallel structure, and we have been developing solution procedures that exploit this in a variety of parallel computing environments.

REFERENCES

1. ADJERID, S., AND FLAHERTY, J. E. A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations. *SIAM J. Numer. Anal. 23* (1986), 778–795.
2. ADJERID, S., AND FLAHERTY, J. E. A moving mesh finite element method with local refinement for parabolic partial differential equations. *Comput. Methods Appl. Mech. Eng. 56* (1986), 3–26.
3. ADJERID, S., AND FLAHERTY, J. E. A local refinement finite element method for two-dimensional parabolic systems. Tech. Rep. 86-7, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, N.Y., 1986.
4. ARNEY, D. C., BISWAS, R., AND FLAHERTY, J. E. A posteriori error estimation of adaptive finite difference schemes for hyperbolic systems. In *Transactions of the Fifth Army Conference on Applied Mathematics and Computing* (West Point, N.Y., June 1987). U.S. Army Research Office, Research Triangle Park, NC, 437–458.
5. ARNEY, D. C., AND FLAHERTY, J. E. A two-dimensional mesh moving technique for time dependent partial differential equations. *J. Comput. Phys. 67* (1986), 124–144.
6. ARNEY, D. C., AND FLAHERTY, J. E. An adaptive local mesh refinement method for time-dependent partial differential equations. Tech. Rep. 86-10, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, N.Y., 1986.
7. ARNEY, D. C., AND FLAHERTY, J. E. An adaptive method with mesh moving and local mesh refinement for time-dependent partial differential equations. In *Transactions of the Fourth Army Conference on Applied Mathematics and Computing* (Ithaca, N.Y., May 1986). U.S. Army Research Office, Research Triangle Park, NC, 1115–1141.
8. BABUSKA, I., CHANDRA, J., AND FLAHERTY, J. E., EDS. *Adaptive Computational Methods for Partial Differential Equations.* SIAM, Philadelphia, Pa., 1983.
9. BABUSKA, I., ZIENKIEWICZ, O. C., GAGO, J. R., AND DE A. OLIVERA, E. R., EDS. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations.* John Wiley, Chichester, 1986.
10. BEN-DOR, G., AND GLASS, I. I. Non-stationary oblique shock-wave reflections: Actual isopycnics and numerical experiments. *AIAA J. 16* (1978), 1146–1153.
11. BEN-DOR, G., AND GLASS, I. I. Domains and boundaries of non-stationary oblique shock-wave reflections. 1. Diatomic gas. *J. Fluid Mech. 92* (1979), 459–496.
12. BERGER, M. On conservation at grid interfaces. ICASE Rep. 84-43, ICASE, NASA Langley Research Center, Hampton, 1984.
13. BERGER, M., AND COLLELA, P. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys. 82* (1989), 64–84.
14. BERGER, M., AND OLIGER, J. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys. 53* (1984), 484–512.
15. BIETERMAN, M., FLAHERTY, J. E., AND MOORE, P. K. Adaptive refinement methods for nonlinear parabolic partial differential equations. In *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, chap. 19, I. Babuska, O. C. Zienkiewicz, J. R. Gago, and E. R. de A. Olivera, Eds., John Wiley, Chichester, 1986.
16. BRACKBILL, J. U., AND SALTZMAN, J. S. Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys. 46* (1982), 342–368.
17. CHAKRAVARTHY, S. R., AND OSHER, S. Computing with high-resolution upwind schemes for hyperbolic equations. In *Large-Scale Computations in Fluid Mechanics, Lectures in Applied*

Mathematics, 22-1, B. E. Engquist, S. Osher, and R. C. J. Somerville, Eds., AMS, Providence, R.I., 1985, 57–86.

18. COYLE, J. M., FLAHERTY, J. E., AND LUDWIG, R. On the stability of mesh equidistribution strategies for time-dependent partial differential equations. *J. Comput. Phys. 62* (1986), 26–39.
19. DAVIS, S. F. TVD finite difference schemes and artificial viscosity. ICASE Rep. 84-20, NASA CR 172373, ICASE, NASA Langley Research Center, Hampton, 1984.
20. DAVIS, S. F., AND FLAHERTY, J. E. An adaptive finite element method for initial-boundary value problems for partial differential equations. *SIAM J. Sci. Stat. Comput. 3* (1982), 6–27.
21. DORR, M. R. The approximation theory for the p-version of the finite element method, I. *SIAM J. Numer. Anal. 21* (1984), 1180–1207.
22. GROPP, W. D. Local uniform mesh refinement with moving grids. *SIAM J. Sci. Stat. Comput. 8* (1987), 292–304.
23. HINDMAN, R. Generalized coordinate forms of governing fluid equations and associated geometrically induced errors. *AIAA J. 20* (1982), 1359–1367.
24. MACCORMACK, R. W. The effect of viscosity in hypervelocity impact cratering. AIAA Paper 69-354, 1969.
25. MCRAE, G., GOODIN, W., AND SEINFELD, J. Numerical solution of the atmospheric diffusion equation for chemically reacting flows. *J. Comput. Phys. 45* (1982), 1–42.
26. ODEN, J. T., STROUBOULIS, T., AND DEVLOO, P. Adaptive finite element methods for the analysis of inviscid compressible flow, I. Fast refinement/unrefinement and moving mesh methods for unstructured meshes. *Comput. Methods Appl. Mech. Eng. 59* (1986), 327–362.
27. RAI, M. M. Patched-grid calculations with the Euler and Navier-Stokes equations. SIAM National Meeting, Boston, Mass., July 1986.
28. RAI, M. M., AND ANDERSON, D. Grid evolution in time asymptotic problems. *J. Comput. Phys. 43* (1981), 327–344.
29. THOMPSON, J. F. A survey of dynamically-adaptive grids in numerical solution of partial differential equations. *Appl. Numer. Math. 1* (1985), 3–27.
30. WOODWARD, P., AND COLLELA, P. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comput. Phys. 54* (1984), 115–173.

Received December 1987; revised April 1989; accepted May 1989